**实验8使用RNN 生成简单序列**

1. **实验内容**

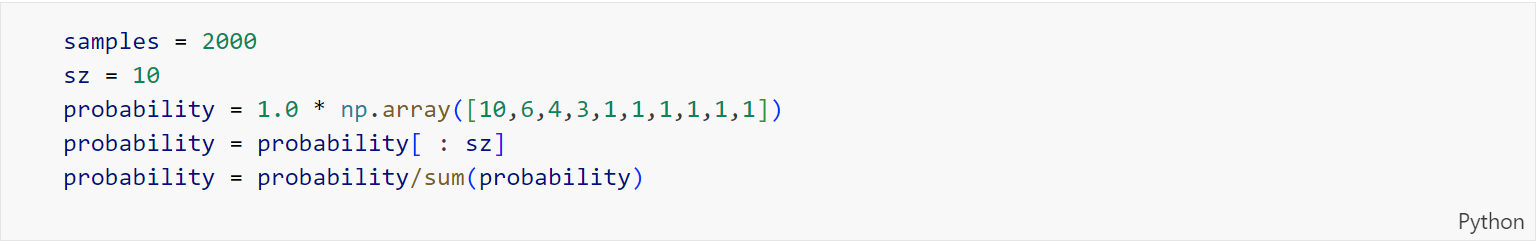
实验内容：首先让神经网络模型学习形如0^n1^n0n1n 形式的上下文无关语法。然后再让模型尝试去生成这样的字符串。在实验的流程中将演示RNN 及LSTM 相关函数的使用方法。探究什么是上下文无关文法、使用RNN 或LSTM 模型生成简单序列的方法、探究RNN 记忆功能的内部原理

实验环境：Python-3.11;PyTorch-cu121;CUDA-12.1;VS code/JupyterNotebook

1. **实验过程和结果**
2. **引入相关包**

****

1. **生成训练数据集**

****

这段代码首先定义一些超参数，代码中sample代表训练集数据的数量，sz代表训练集中01字符的最大长度为10个，针对01个数的取值n设定其取值从1-10的可能性权重数组possibility，经过系列处理成为n取值各值得可能性，被用于np.random.choice的参数中。

****

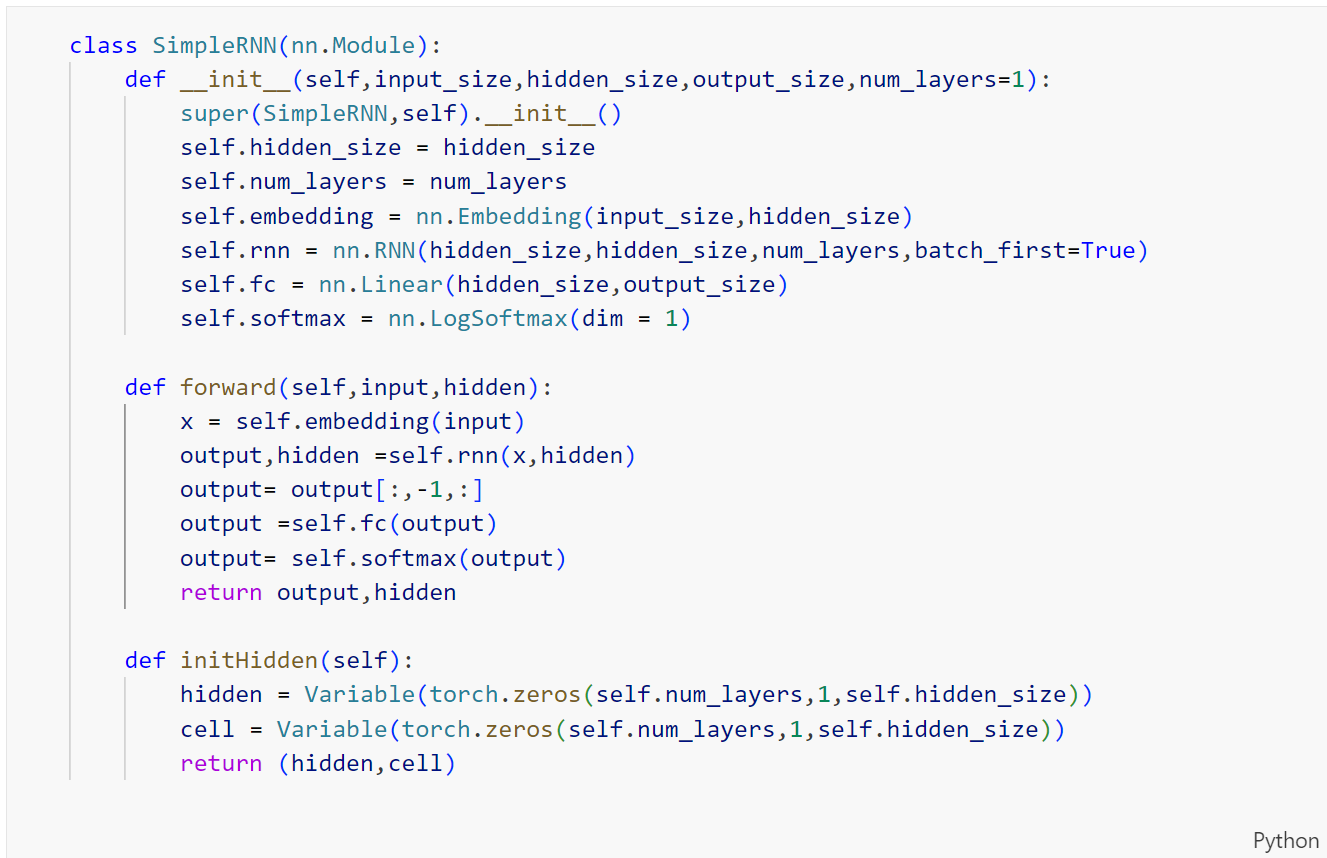
****

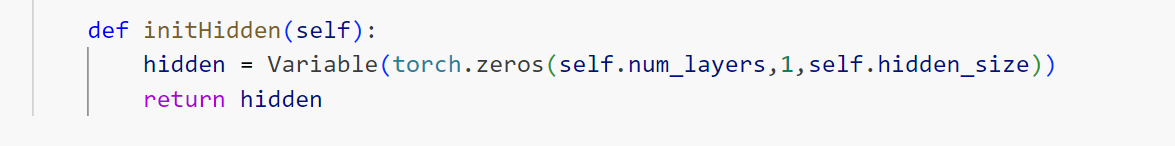
这两段代码结构相同，分别用于生成训练数据集和校验数据集，不同处为校验集大小只有训练集的约1/10，首先根据概率选取01个数n，运用列表操作在01串前插入3，后追加2，形成一条数据再加入到数据集中。

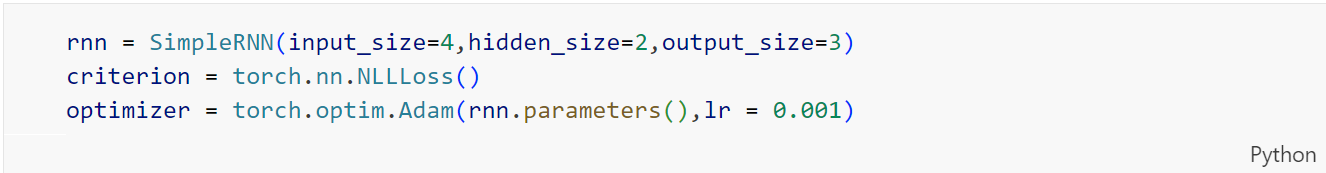


这段代码在先前校验集基础上追加了两条01长度为11和12的数据，并将校验集顺序打乱。

1. **定义RNN模型**

****

****

****

定义RNN类的方法与前面的实验大致相同，按层的顺序首先是embedding层将一串字符映射为一个n维向量，将数据变为hidden\_size的长度，代替了onehot编码，embedding层存在可学习参数，在训练的过程中学习到最佳的权重；其次是经过封装的RNN层，经过嵌入后rnn输入尺寸和隐层尺寸相同，rnn的层数为1，最后经过一个全连接层，全连接层的输出神经元个数在此例中为3，代表012，2代表的是字符串的结束，输出经过softmax处理后变为三种输出的预测概率。

然后对SimpleRNN进行实例化，定义输入隐层输出层的大小。损失函数采用NLLloss负对数似然损失函数，经过查阅，NLLLoss函数配合softmax使用和交叉熵损失函数是等价的。RNN类中还有initHidden函数，其作用是对隐含单元进行初始化。

前向forward()函数传入的参数既有input，也有hidden表示上一时刻或者初始时刻的隐含单元输出值。其中x的尺寸是(length\_seq,batch\_size,

input\_size），此时第一个维度大于1时RNN就会多步运行，output包含每一步运行的结果，hidden只包含最后一步运行隐含单元的输出。对output取[:,-1,:]代表获取最后一个时刻的隐含层输出作为这一层的输出

****

****

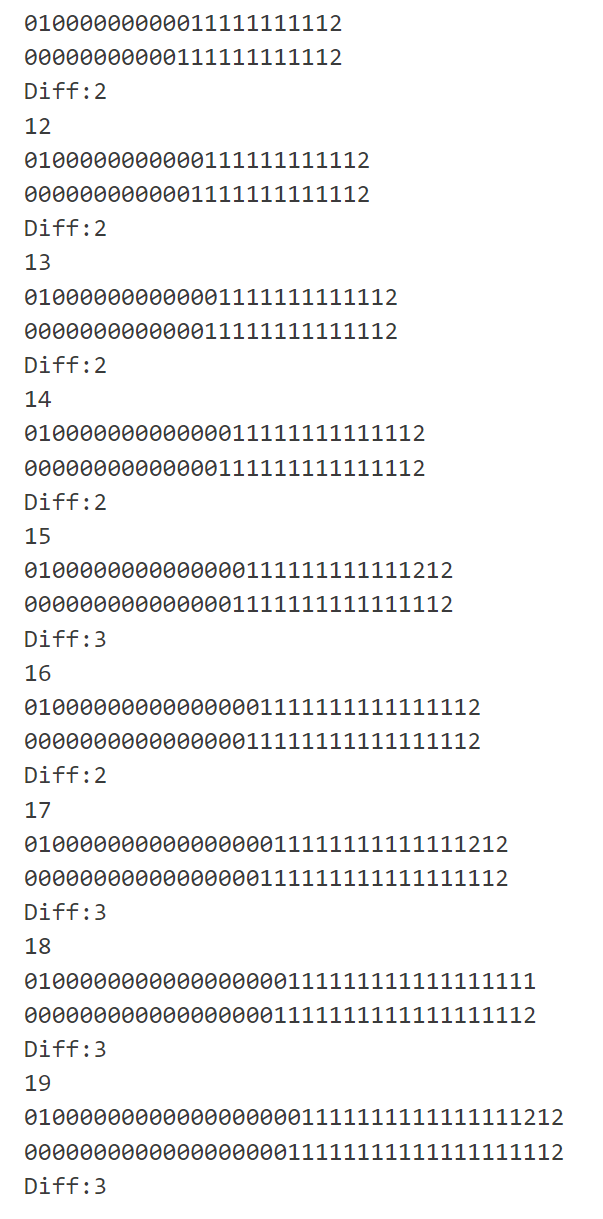
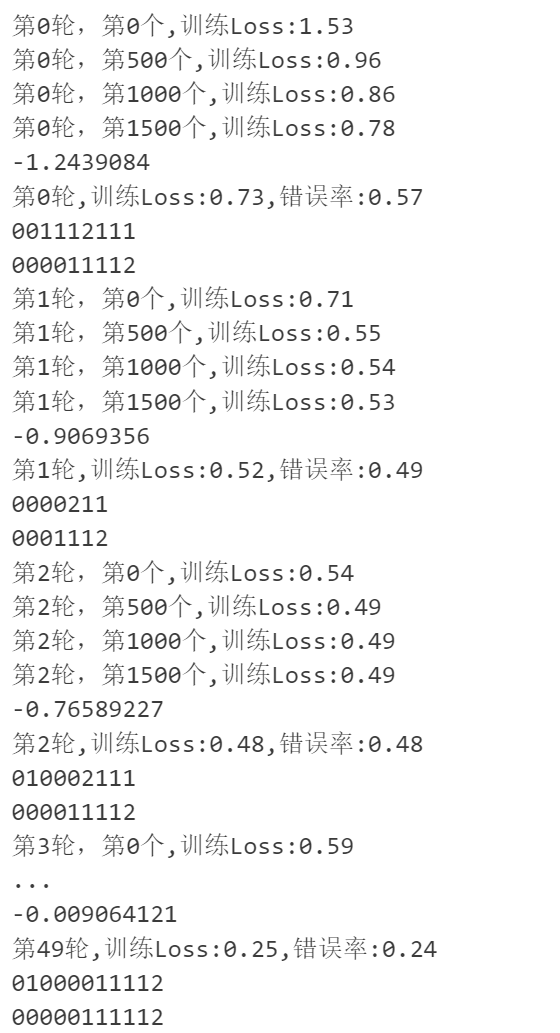
1. **训练RNN模型**



进行RNN 模型的训练。在下面的训练代码中实际上进行了三重循环，Epoch 作为第一重循环，然后在trainRNN 中对每个train\_set 中的字符串做第二重循环，最后是对每一个字符串中的每一个字符做循环。

1. **观察RNN模型的学习结果**

****

****

起初训练轮次设置为20，训练没有达到理想效果，且每次训练的效果差异很大。于是将训练轮次改为50，训练正确率得到提高。但可以发现训练出来的模型也不是最理想的效果，经过多次尝试，训练出的结果如上图所示，但总是在第二个符号处出错，可能是陷入了某个局部最优解中。

对于RNN来说，通过观察学习结果可以发现当n比较小时RNN一般只在0变为1时犯错，在n=15时开始出现其他错误，因此可以得到结论该大小的RNN的记忆容量约为14。

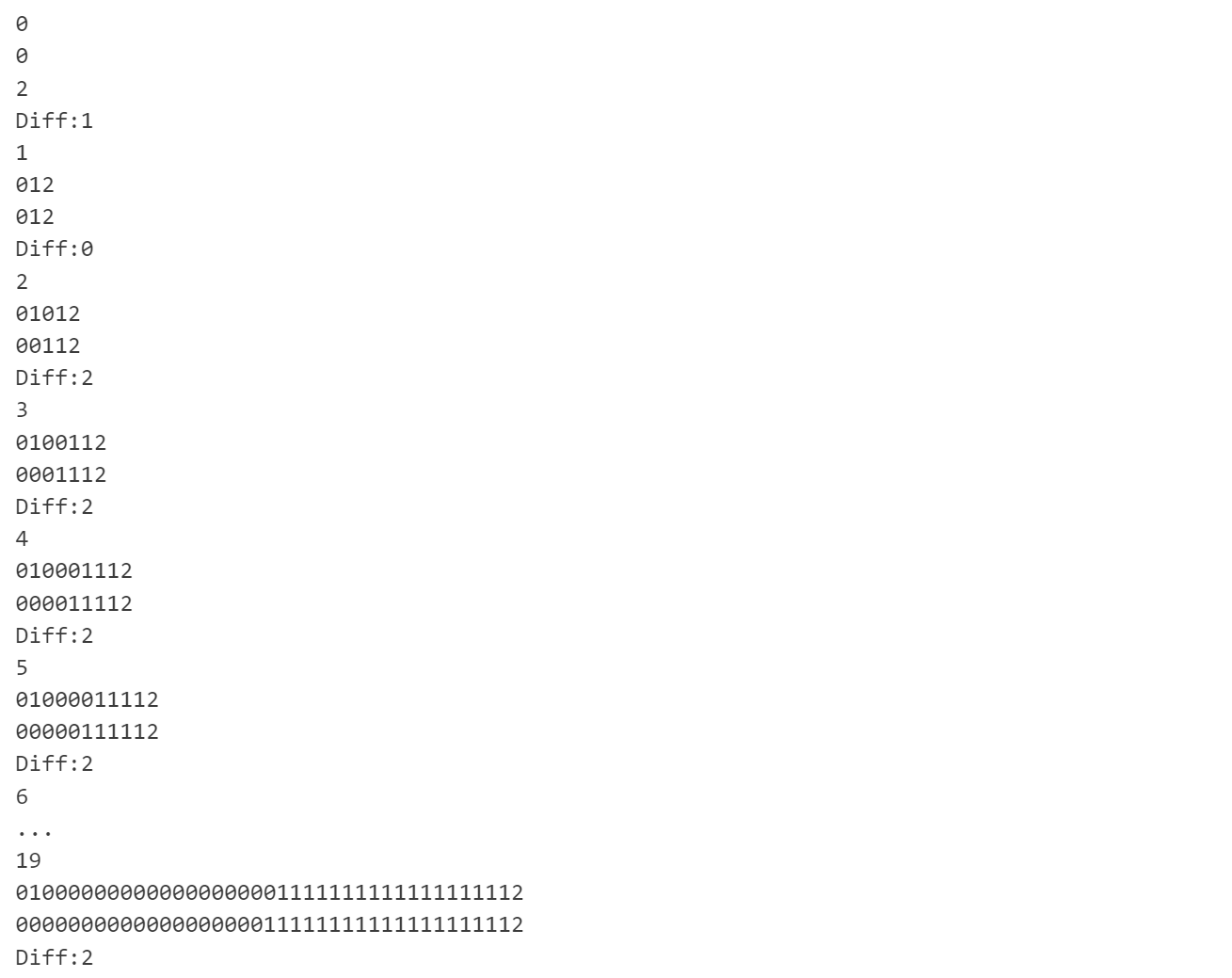
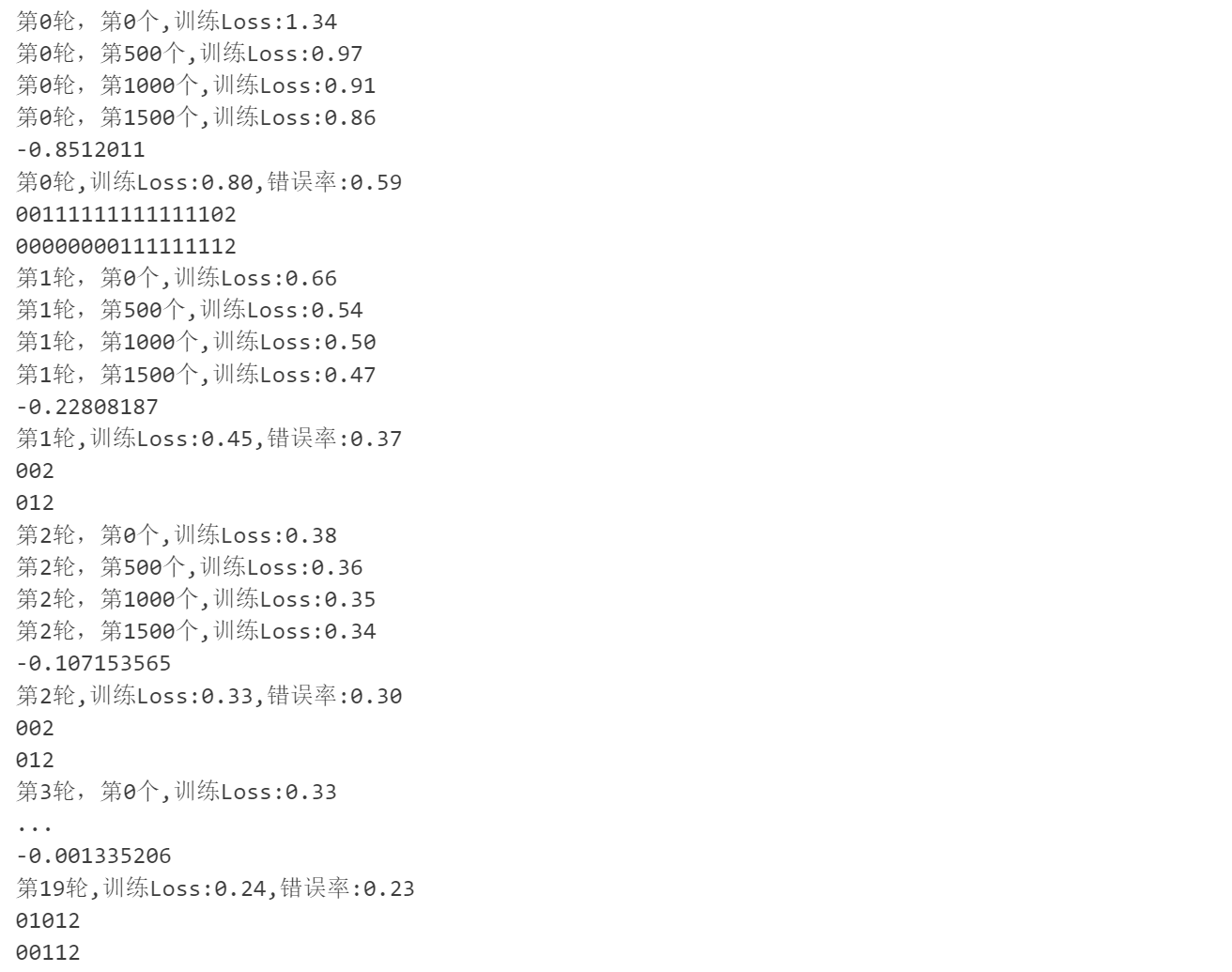
1. **实现LSTM模型**

****

****

****

****

****

LSTM 模型的代码与RNN 几乎相同，只有在初始化隐藏层状态的时候，LSTM 除了初始化隐藏层的状态，还初始化了隐含层内部细胞的状态，也就是各个“门控单元”的状态。通过观察相同尺寸的RNN和LSTM模型的训练效果，LSTM具有更长的记忆能力。通过观察结果LSTM模型即使n达到了19依然能够保持较好的错误率，验证了LSTM具有更长的记忆能力。而LSTM只训练了20个轮次。

1. **心得体会**

本次实验通过生成了简单的序列练习使用了RNN和LSTM。两者间进行对比，两者的基本代码相同，LSTM具有更长的记忆能力。RNN结构在前馈神经网络的基础上令隐层的输出和前一次的输出值相关，增加了隐层内部的连接赋予了RNN一定的记忆能力。

LSTM相比RNN增加了几个内部的门结构，通过控制门结构使遗忘门被输入信号控制，加入内部储存提高了LSTM的长期记忆能力。

实验过程中RNN训练函数中出现报错，rnn中不能传入元组类型的变量，变量的大小不匹配，有关hidden的维度出现错误，经过如下的判断语句后再输入到模型中问题解决。

hidden = hidden[0] if isinstance(hidden, tuple) else hidden

改正：通过复习过程中仔细阅读代码，读懂每一句的作用和含义帮助我更好地记忆各种网络模型地结构，也在这个过程中找到了先前没有发现的错误。RNN和LSTM两个模型代码的主要区别在于initHidden函数中初始化的隐层不同，在实验过程中没有阅读教材，而是按照实验指导书中有关LSTM的代码直接修改，两者使用了相同的类定义代码，但RNN中没有cell结构，因此initHidden的返回值hidden作为参数传入rnn中时出现报错，删除RNN中对cell的初始化和返回值后就不需要上述判断语句了。